```
function x = f(n)
    x = 1;
    for i = 1:n
        for j = 1:n
            x = x + 1;
```

1 Ans:- Inner loop:-

It executes n times for each phase in outer loop.

∴ n executions for inner loop

Outer loop:-

It executes n times from i=1 to i=n

∴ n executions for outer loop

$$\text{Total executions} = \sum_{i=1}^{n} \sum_{j=1}^{n} 1$$

$$= \sum_{i=1}^{n} n$$

$$= n \times n$$

$$= n^2$$

∴ The total runtime of the algo is $O(n^2)$

3 Ans:- Big-O-Notation (Upper bound)

$O(n^2)$ - function does not grow faster than quadratic

Big-Omega

$\Omega(n^2)$ - function does not grow slower than quadratic

Big-Theta

$\Theta(n^2)$ - function grows asymptotically as n^2

4 Ans:- $y = i * j$, actual time taken by the modified function will be slightly longer due to it.

$\therefore O(n^2)$ is the time complexity.

5 Ans:- It doesn't effect much in the time complexity.